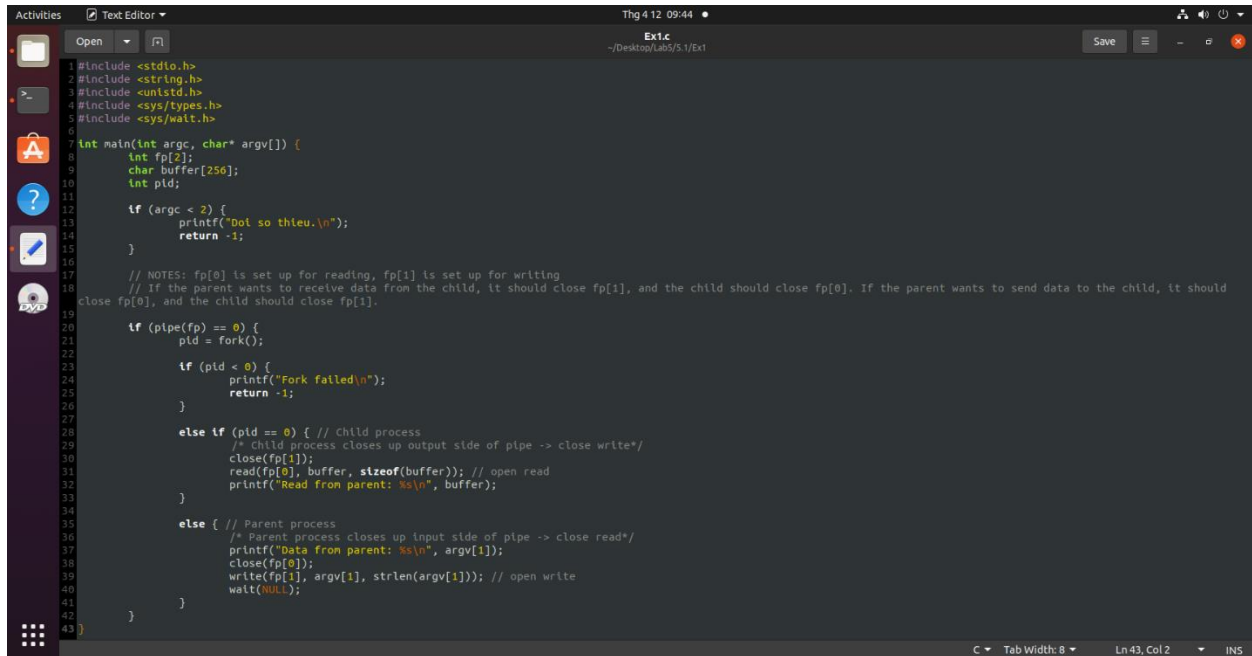


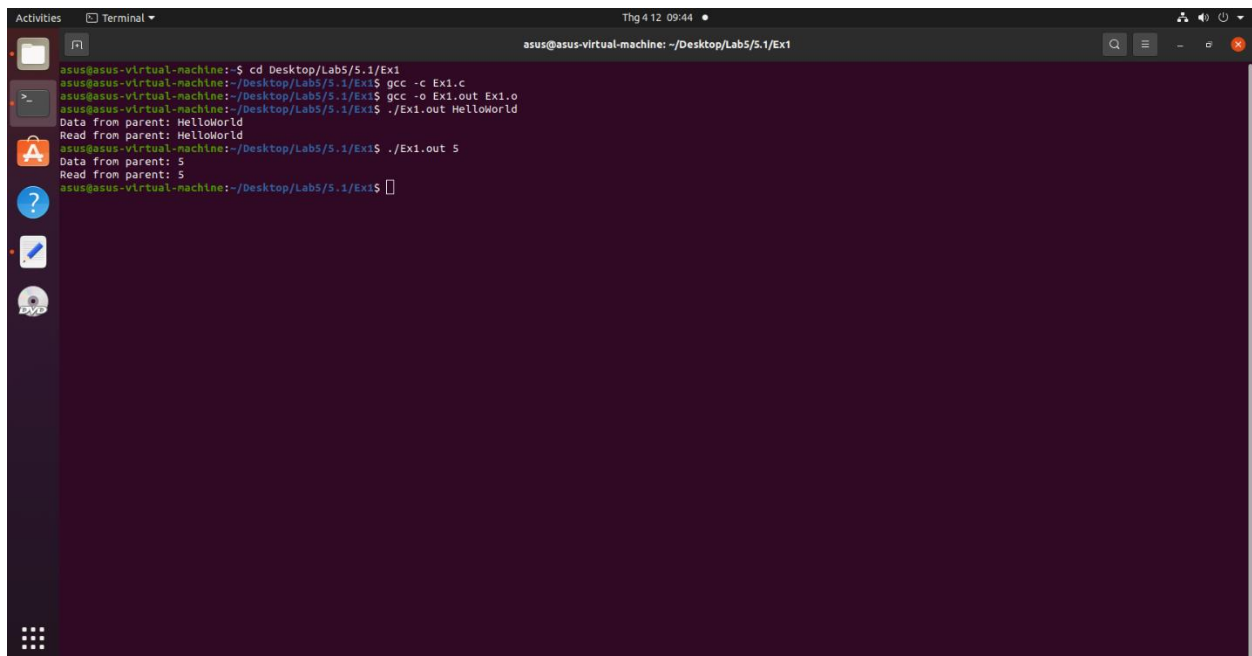
LAB 5

5.1 – Đường ống liên lạc

Bài 1:

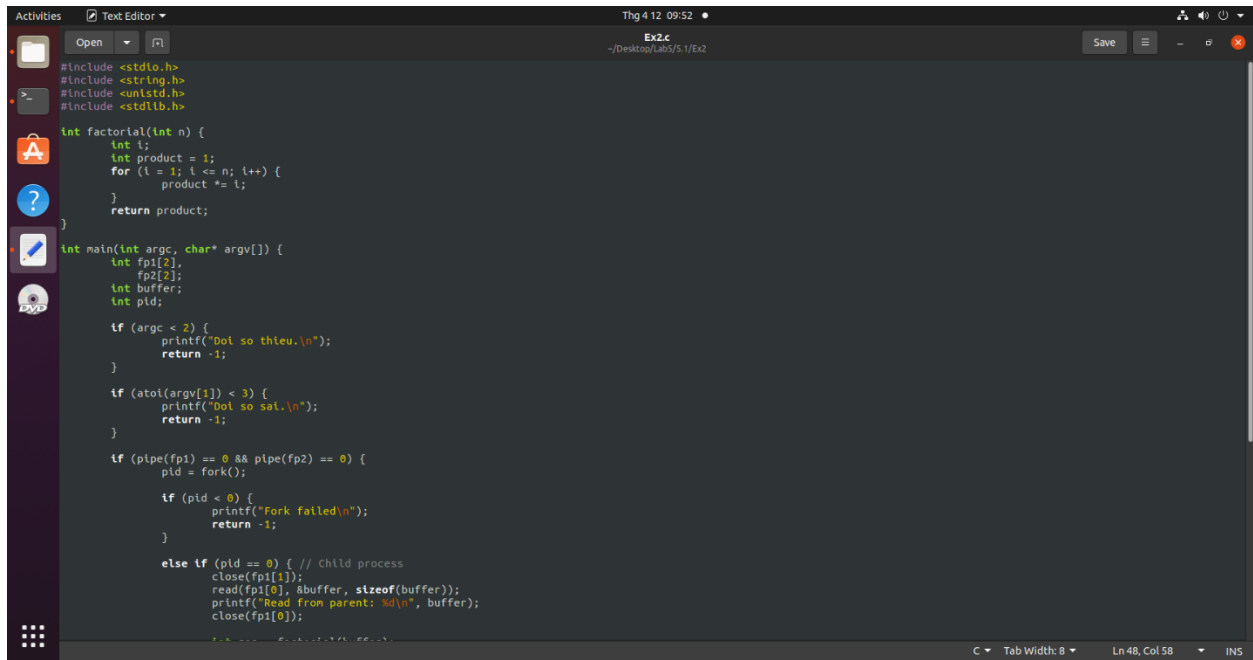


```
1#include <stdio.h>
2#include <string.h>
3#include <unistd.h>
4#include <sys/types.h>
5#include <sys/wait.h>
6
7int main(int argc, char* argv[]) {
8    int fp[2];
9    char buffer[256];
10    int pld;
11
12    if (argc < 2) {
13        printf("Dot so thieu.\n");
14        return -1;
15    }
16
17    // NOTES: fp[0] is set up for reading, fp[1] is set up for writing
18    // If the parent wants to receive data from the child, it should close fp[1], and the child should close fp[0]. If the parent wants to send data to the child, it should
19    close fp[0], and the child should close fp[1].
20
21    if (pipe(fp) == 0) {
22        pld = fork();
23
24        if (pld < 0) {
25            printf("Fork failed\n");
26            return -1;
27        }
28
29        else if (pld == 0) { // Child process
30            /* child process closes up output side of pipe -> close write*/
31            close(fp[1]);
32            read(fp[0], buffer, sizeof(buffer)); // open read
33            printf("Read from parent: %s\n", buffer);
34        }
35
36        else { // Parent process
37            /* Parent process closes up input side of pipe -> close read*/
38            printf("Data from parent: %s\n", argv[1]);
39            close(fp[0]);
40            write(fp[1], argv[1], strlen(argv[1])); // open write
41            wait(NULL);
42        }
43    }
44}
```



```
asus@asus-virtual-machine: ~/Desktop/Lab5/5.1/Ex1
asus@asus-virtual-machine:~$ cd Desktop/Lab5/5.1/Ex1
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex1$ gcc -c Ex1.c
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex1$ gcc -o Ex1.out Ex1.o
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex1$ ./Ex1.out HelloWorld
Data from parent: HelloWorld
Read from parent: HelloWorld
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex1$ ./Ex1.out 5
Data from parent: 5
Read from parent: 5
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex1$
```

Bài 2:



```
Activities Text Editor Thg 4 12 09:52
Ex2c
~/Desktop/Lab5/5.1/Ex2 Save

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>

int factorial(int n) {
    int i;
    int product = 1;
    for (i = 1; i <= n; i++) {
        product *= i;
    }
    return product;
}

int main(int argc, char* argv[]) {
    int fp1[2];
    int fp2[2];
    int buffer;
    int pid;

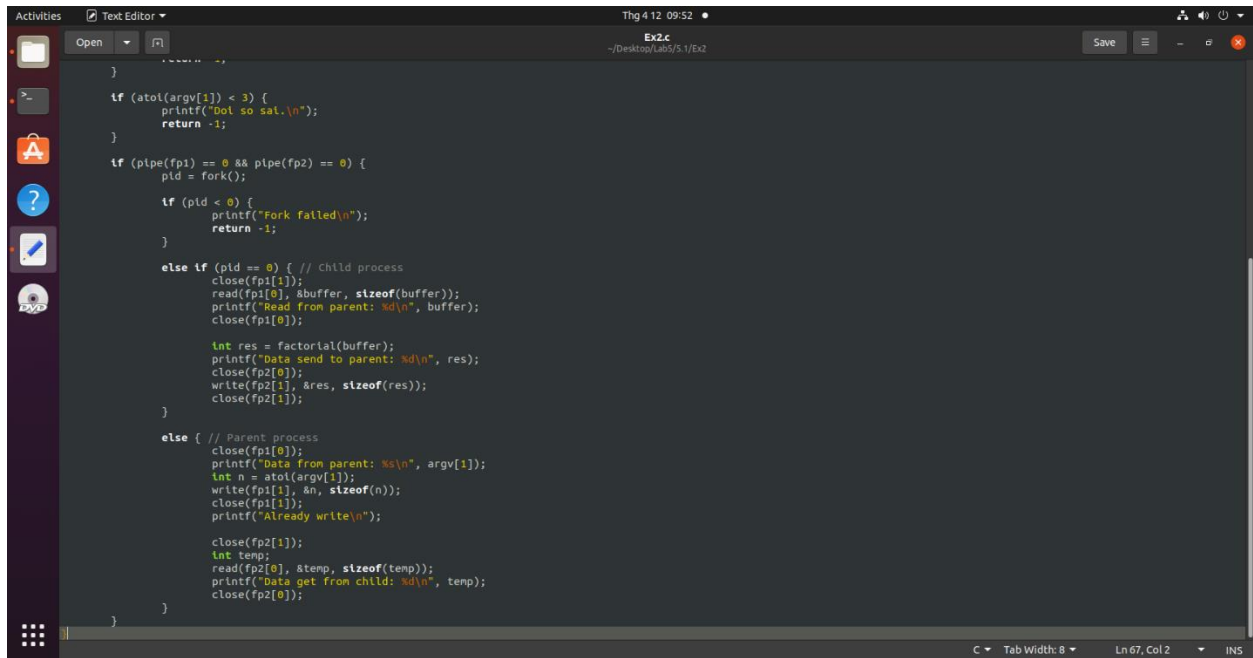
    if (argc < 2) {
        printf("Doi so thieu.\n");
        return -1;
    }

    if (atoi(argv[1]) < 3) {
        printf("Doi so sai.\n");
        return -1;
    }

    if (pipe(fp1) == 0 && pipe(fp2) == 0) {
        pid = fork();

        if (pid < 0) {
            printf("Fork failed\n");
            return -1;
        }

        else if (pid == 0) { // Child process
            close(fp1[1]);
            read(fp1[0], &buffer, sizeof(buffer));
            printf("Read from parent: %d\n", buffer);
            close(fp1[0]);
```



```
        }

        if (atoi(argv[1]) < 3) {
            printf("Doi so sai.\n");
            return -1;
        }

        if (pipe(fp1) == 0 && pipe(fp2) == 0) {
            pid = fork();

            if (pid < 0) {
                printf("Fork failed\n");
                return -1;
            }

            else if (pid == 0) { // Child process
                close(fp1[1]);
                read(fp1[0], &buffer, sizeof(buffer));
                printf("Read from parent: %d\n", buffer);
                close(fp1[0]);

                int res = factorial(buffer);
                printf("Data send to parent: %d\n", res);
                close(fp2[0]);
                write(fp2[1], &res, sizeof(res));
                close(fp2[1]);
            }

            else { // Parent process
                close(fp1[0]);
                printf("Data from parent: %s\n", argv[1]);
                int n = atoi(argv[1]);
                write(fp1[1], &n, sizeof(n));
                close(fp1[1]);
                printf("Already write\n");

                close(fp2[1]);
                int temp;
                read(fp2[0], &temp, sizeof(temp));
                printf("Data get from child: %d\n", temp);
                close(fp2[0]);
            }
        }
    }
}
```

```
Activities Terminal Thg 4 12 09:52
asus@asus-virtual-machine: ~/Desktop/Lab5/5.1/Ex2
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex2$ cd -
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex2$ gcc -c Ex2.c
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex2$ gcc -o Ex2.out Ex2.o
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex2$ ./Ex2.out 4
Data from parent: 4
Already write
Read from parent: 4
Data send to parent: 24
Data get from child: 24
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex2$ ./Ex2.out 5
Data from parent: 5
Already write
Read from parent: 5
Data send to parent: 120
Data get from child: 120
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex2$ ./Ex2.out 10
Data from parent: 10
Already write
Read from parent: 10
Data send to parent: 3628800
Data get from child: 3628800
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex2$
```

Bài 3:

```
Activities Text Editor Thg 4 12 11:24
Ex3.c
~/Desktop/Lab5/5.1/Ex3
Save

1#include <stdio.h>
2#include <string.h>
3#include <unistd.h>
4#include <stdlib.h>
5#include <sys/types.h>
6#include <sys/wait.h>
7
8int caculator(int a, int b, char operator) {
9    switch(operator) {
10        case '+':
11            return a + b;
12        case '-':
13            return a - b;
14        case '*':
15            return a * b;
16        case '/':
17            return a / b;
18    }
19}
20
21int main(int argc, char* argv[]) {
22    int fp1[2],
23        fp2[2],
24        fp3[2],
25        fp4[2],
26        a,
27        b;
28    char operator;
29    int pid;
30
31    if (argc < 4) {
32        printf("Dot so thieu.\n");
33        return -1;
34    }
35
36    if (pipe(fp1) == 0 && pipe(fp2) == 0 && pipe(fp3) == 0 && pipe(fp4) == 0) {
37        pid = fork();
38
39        if (pid < 0) {
40            printf("Fork failed\n");
41            return -1;
42        }
43
44        else if (pid == 0) { // Child process
```

```
Activities Text Editor Thg 4 12 11:24
Ex3.c
~/Desktop/Lab5/5.1/Ex3
Save

42    }
43
44    else if (pid == 0) { // Child process
45        close(fp1[1]);
46        read(fp1[0], &a, sizeof(a));
47        close(fp1[0]);
48
49        close(fp2[1]);
50        read(fp2[0], &b, sizeof(b));
51        close(fp2[0]);
52
53        close(fp3[1]);
54        read(fp3[0], &operator, sizeof(operator));
55        close(fp3[0]);
56
57        int res = caculator(a, b, operator);
58        close(fp4[0]);
59        write(fp4[1], &res, sizeof(res));
60        close(fp4[1]);
61    }
62
63    else { // Parent process
64        close(fp1[0]);
65        int c = atoi(argv[1]);
66        write(fp1[1], &c, sizeof(c));
67        close(fp1[1]);
68
69        close(fp2[0]);
70        int d = atoi(argv[2]);
71        write(fp2[1], &d, sizeof(d));
72        close(fp2[1]);
73
74        close(fp3[0]);
75        char tempString[1];
76        strcpy(tempString, argv[3]);
77        char tempOperator = tempString[0];
78        write(fp3[1], &tempOperator, sizeof(tempOperator));
79        close(fp3[1]);
80
81        close(fp4[1]);
82
83        int temp;
84        read(fp4[0], &temp, sizeof(temp));
85        close(fp4[0]);
86    }
```

```
Activities Text Editor Thg 4 12 11:24
Ex3.c
~/Desktop/Lab5/5.1/Ex3

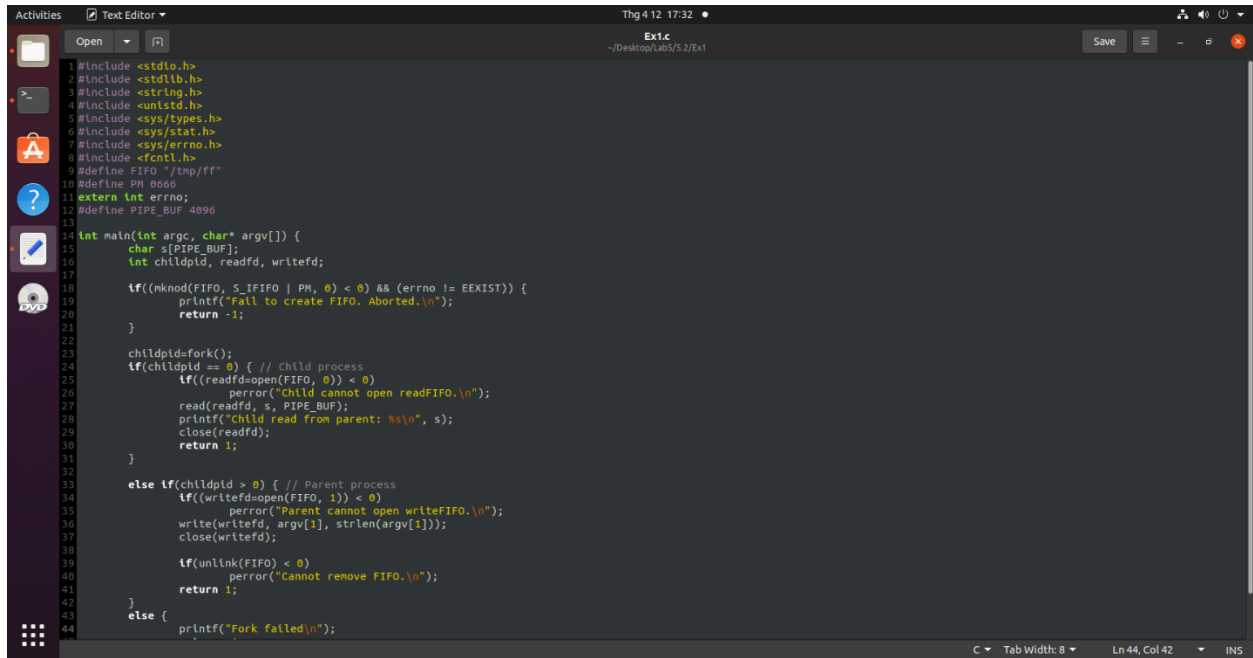
47 close(fp2[0]);
48 close(fp1[0]);
49 close(fp2[1]);
50 read(fp2[0], &b, sizeof(b));
51 close(fp2[0]);
52
53 close(fp3[1]);
54 read(fp3[0], &operator, sizeof(operator));
55 close(fp3[0]);
56
57 int res = calculator(a, b, operator);
58 close(fp4[0]);
59 write(fp4[1], &res, sizeof(res));
60 close(fp4[1]);
61
62
63 else { // Parent process
64 close(fp1[0]);
65 int c = atoi(argv[1]);
66 write(fp1[1], &c, sizeof(c));
67 close(fp1[1]);
68
69 close(fp2[0]);
70 int d = atoi(argv[2]);
71 write(fp2[1], &d, sizeof(d));
72 close(fp2[1]);
73
74 close(fp3[0]);
75 char tempString[1];
76 strcpy(tempString, argv[3]);
77 char tempOperator = tempString[0];
78 write(fp3[1], &tempOperator, sizeof(tempOperator));
79 close(fp3[1]);
80
81
82 close(fp4[1]);
83 int temp;
84 read(fp4[0], &temp, sizeof(temp));
85 close(fp4[0]);
86
87 printf("kd %c kd = %d\n", c, tempOperator, d, temp);
88
89
90 }
```

```
Activities Terminal Thg 4 12 11:24
asus@asus-virtual-machine: ~/Desktop/Lab5/5.1/Ex3

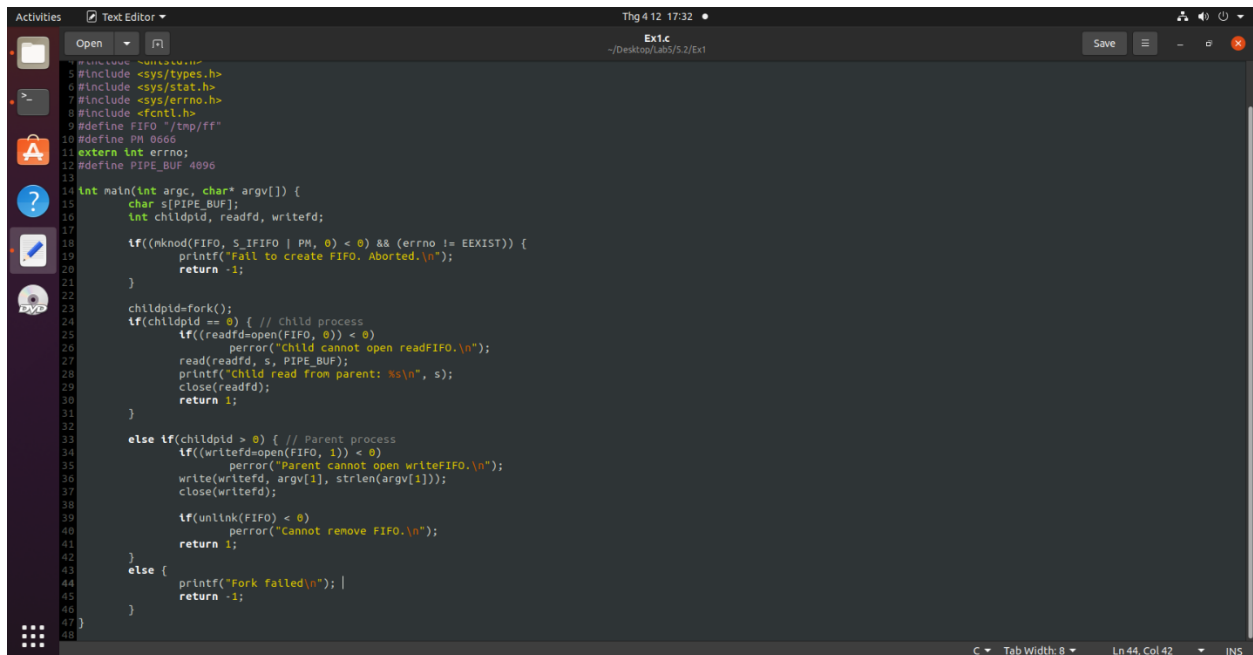
asus@asus-virtual-machine:~$ cd Desktop/Lab5/5.1/Ex3
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex3$ gcc -c Ex3.c
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex3$ gcc -o Ex3.out Ex3.o
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex3$ ./Ex3.out 6 4 +
6 + 4 = 10
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex3$ ./Ex3.out 6 4 -
6 - 4 = 2
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex3$ ./Ex3.out 6 4 \*
6 * 4 = 24
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex3$ ./Ex3.out 6 4 /
6 / 4 = 1
asus@asus-virtual-machine:~/Desktop/Lab5/5.1/Ex3$
```

5.2 – Đường ống được đặt tên

Bài 1:



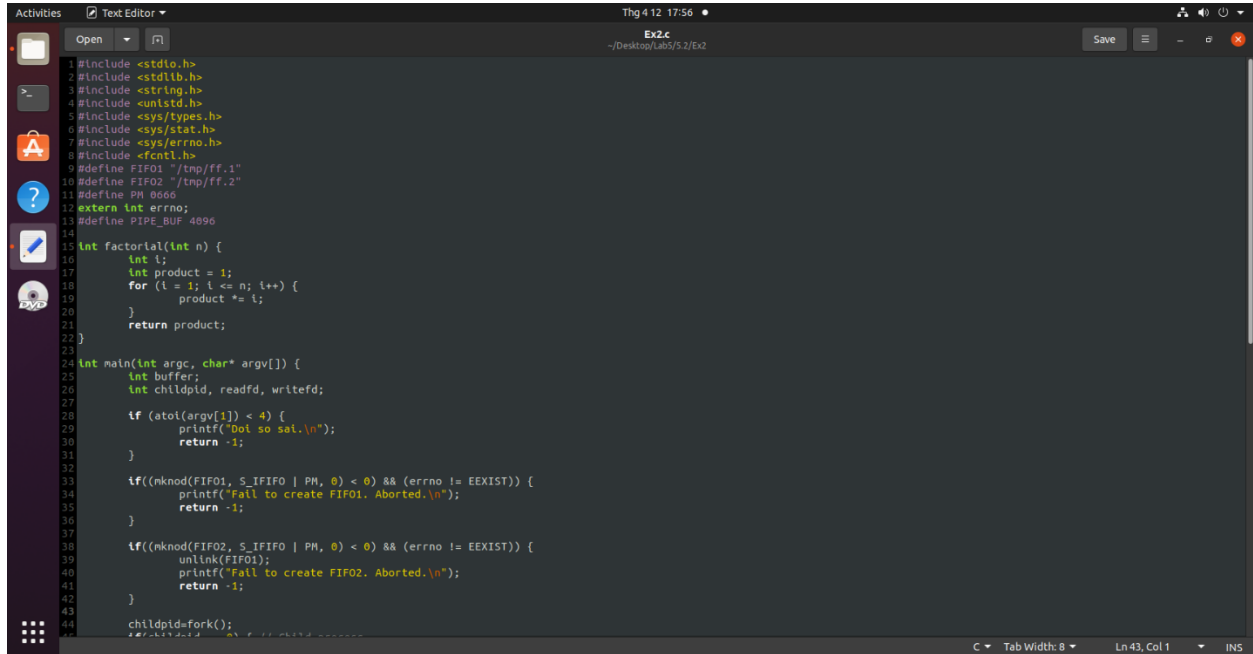
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 #include <sys/stat.h>
7 #include <sys/errno.h>
8 #include <fcntl.h>
9 #define FIFO "/tmp/ff"
10 #define PM 0666
11 extern int errno;
12 #define PIPE_BUF 4096
13
14 int main(int argc, char* argv[]) {
15     char s[PIPE_BUF];
16     int chldpid, readfd, writefd;
17
18     if((mknod(FIFO, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
19         printf("Fail to create FIFO. Aborted.\n");
20         return -1;
21     }
22
23     chldpid=fork();
24     if(chldpid == 0) { // Child process
25         if((readfd=open(FIFO, 0)) < 0)
26             perror("Child cannot open readFIFO.\n");
27         read(readfd, s, PIPE_BUF);
28         printf("Child read from parent: %s\n", s);
29         close(readfd);
30         return 1;
31     }
32
33     else if(chldpid > 0) { // Parent process
34         if((writefd=open(FIFO, 1)) < 0)
35             perror("Parent cannot open writeFIFO.\n");
36         write(writefd, argv[1], strlen(argv[1]));
37         close(writefd);
38
39         if(unlink(FIFO) < 0)
40             perror("Cannot remove FIFO.\n");
41         return 1;
42     }
43     else {
44         printf("Fork failed\n");
```



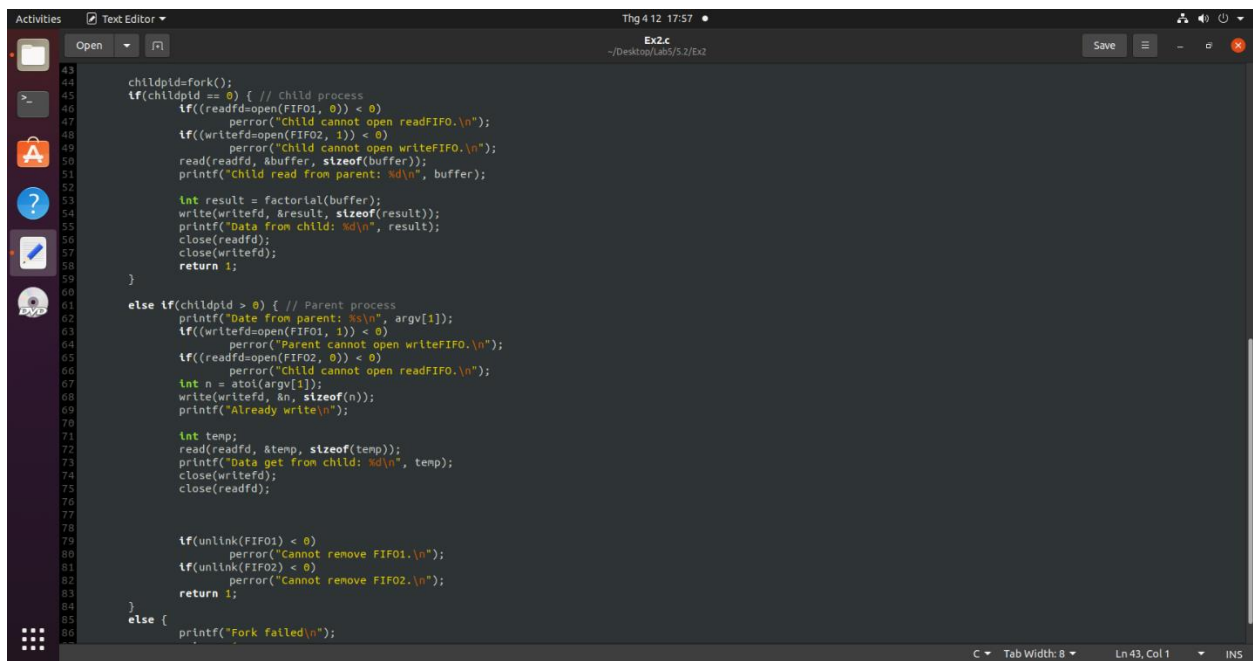
```
44         printf("Fork failed\n");
45         return -1;
46     }
47 }
48
```

```
Activities Terminal Thg 4 12 17:32
asus@asus-virtual-machine: ~/Desktop/Lab5/S.2/Ex1
asus@asus-virtual-machine:~$ cd Desktop/Lab5/S.2/Ex1
asus@asus-virtual-machine:~/Desktop/Lab5/S.2/Ex1$ gcc -c Ex1.c
asus@asus-virtual-machine:~/Desktop/Lab5/S.2/Ex1$ gcc -o Ex1.out Ex1.o
asus@asus-virtual-machine:~/Desktop/Lab5/S.2/Ex1$ ./Ex1.out HelloWorld
Child read from parent: HelloWorld
asus@asus-virtual-machine:~/Desktop/Lab5/S.2/Ex1$ ./Ex1.out Hello
Child read from parent: Hello
asus@asus-virtual-machine:~/Desktop/Lab5/S.2/Ex1$
```

Bài 2:



```
1#include <stdio.h>
2#include <stdlib.h>
3#include <string.h>
4#include <unistd.h>
5#include <sys/types.h>
6#include <sys/stat.h>
7#include <sys/errno.h>
8#include <fcntl.h>
9#define FIFO1 "/tmp/ff.1"
10#define FIFO2 "/tmp/ff.2"
11#define PM 0666
12extern int errno;
13#define PIPE_BUF 4096
14
15int factorial(int n) {
16    int i;
17    int product = 1;
18    for (i = 1; i <= n; i++) {
19        product *= i;
20    }
21    return product;
22}
23
24int main(int argc, char* argv[]) {
25    int buffer;
26    int childpid, readfd, writefd;
27
28    if (atoi(argv[1]) < 4) {
29        printf("Dot so sat.\n");
30        return -1;
31    }
32
33    if((mkfifo(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
34        printf("Fail to create FIFO1. Aborted.\n");
35        return -1;
36    }
37
38    if((mkfifo(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
39        unlink(FIFO1);
40        printf("Fail to create FIFO2. Aborted.\n");
41        return -1;
42    }
43
44    childpid=fork();
45    if (childpid == 0) { // child process
```



```
46    if(readfd=open(FIFO1, 0) < 0)
47        perror("Child cannot open readFIFO.\n");
48    if(writefd=open(FIFO2, 1) < 0)
49        perror("Child cannot open writeFIFO.\n");
50    read(readfd, &buffer, sizeof(buffer));
51    printf("Child read from parent: %d\n", buffer);
52
53    int result = factorial(buffer);
54    write(writefd, &result, sizeof(result));
55    printf("Data from child: %d\n", result);
56    close(readfd);
57    close(writefd);
58    return 1;
59}
60
61else if(childpid > 0) { // Parent process
62    printf("Date from parent: %s\n", argv[1]);
63    if(writefd=open(FIFO1, 1) < 0)
64        perror("Parent cannot open writeFIFO.\n");
65    if(readfd=open(FIFO2, 0) < 0)
66        perror("Child cannot open readFIFO.\n");
67    int n = atoi(argv[1]);
68    write(writefd, &n, sizeof(n));
69    printf("Already write\n");
70
71    int temp;
72    read(readfd, &temp, sizeof(temp));
73    printf("Data get from child: %d\n", temp);
74    close(writefd);
75    close(readfd);
76
77    if(unlink(FIFO1) < 0)
78        perror("Cannot remove FIFO1.\n");
79    if(unlink(FIFO2) < 0)
80        perror("Cannot remove FIFO2.\n");
81    return 1;
82}
83
84else {
85    printf("Fork failed\n");
86}
```



```
Activities Text Editor Thg 4 12 17:57
Ex2.c
~/Desktop/Lab5/5.2/Ex2

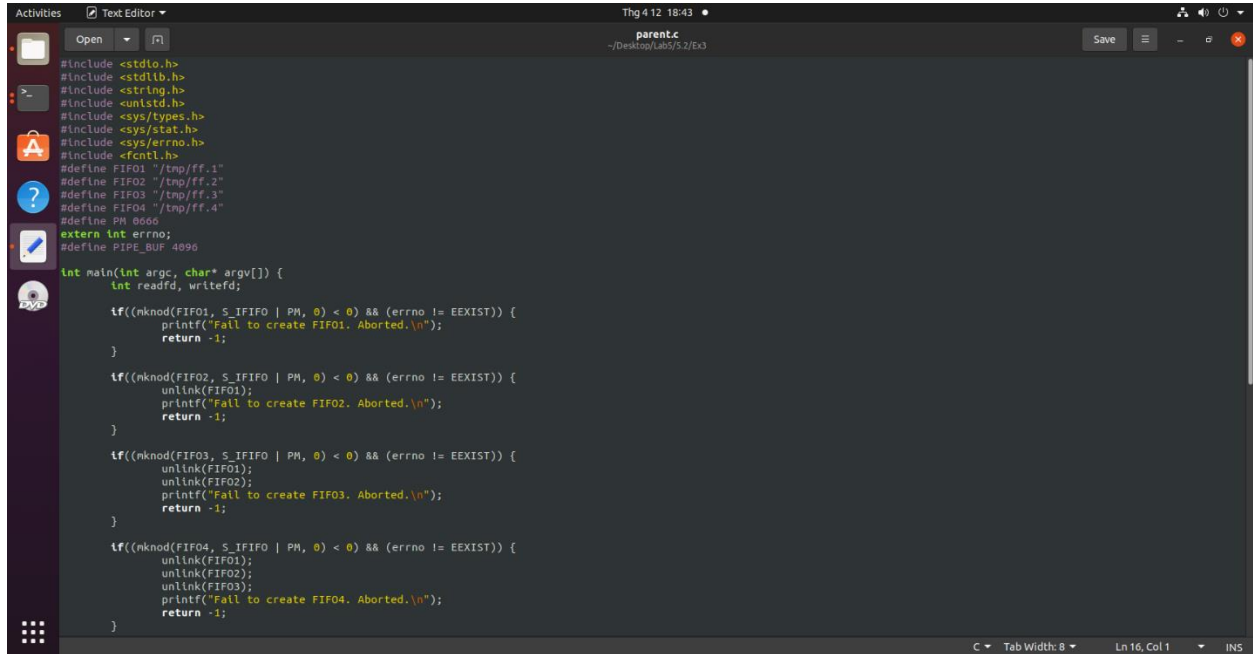
46     if((readfd=open(FIFO1, 0)) < 0)
47         perror("Child cannot open readFIFO.\n");
48     if((writefd=open(FIFO2, 1)) < 0)
49         perror("Child cannot open writeFIFO.\n");
50     read(readfd, &buffer, sizeof(buffer));
51     printf("Child read from parent: %d\n", buffer);
52
53     int result = factorial(buffer);
54     write(writefd, &result, sizeof(result));
55     printf("Data from child: %d\n", result);
56     close(readfd);
57     close(writefd);
58     return 1;
59 }
60
61 else if(childpid > 0) { // Parent process
62     printf("Date from parent: %i\n", argv[1]);
63     if((writefd=open(FIFO1, 1)) < 0)
64         perror("Parent cannot open writeFIFO.\n");
65     if((readfd=open(FIFO2, 0)) < 0)
66         perror("Child cannot open readFIFO.\n");
67     int n = atoi(argv[1]);
68     write(writefd, &n, sizeof(n));
69     printf("Already write\n");
70
71     int temp;
72     read(readfd, &temp, sizeof(temp));
73     printf("Data get from child: %d\n", temp);
74     close(writefd);
75     close(readfd);
76
77     if(unlink(FIFO1) < 0)
78         perror("Cannot remove FIFO1.\n");
79     if(unlink(FIFO2) < 0)
80         perror("Cannot remove FIFO2.\n");
81     return 1;
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }

Bracket match found on line: 24 C Tab Width: 8 Ln 89, Col 2 INS
```

```
Activities Terminal Thg 4 12 17:56
asus@asus-virtual-machine: ~/Desktop/Lab5/5.2/Ex2

asus@asus-virtual-machine:~$ cd Desktop/Lab5/5.2/Ex2
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex2$ gcc -c Ex2.c
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex2$ gcc -o Ex2.out Ex2.o
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex2$ ./Ex2.out 5
Date from parent: 5
Child read from parent: 5
Already write
Data from child: 120
Data get from child: 120
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex2$ ./Ex2.out 10
Date from parent: 10
Already write
Child read from parent: 10
Data from child: 3628800
Data get from child: 3628800
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex2$
```

Bài 3:



```
Activities Text Editor Thg 4 12 18:43
parent.c
~/Desktop/Lab5/S.2/Ex3

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <fcntl.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define FIFO3 "/tmp/ff.3"
#define FIFO4 "/tmp/ff.4"
#define PM 0660
extern int errno;
#define PIPE_BUF 4096

int main(int argc, char* argv[]) {
    int readfd, writefd;

    if((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        printf("Fail to create FIFO1. Aborted.\n");
        return -1;
    }

    if((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        unlink(FIFO1);
        printf("Fail to create FIFO2. Aborted.\n");
        return -1;
    }

    if((mknod(FIFO3, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        unlink(FIFO1);
        unlink(FIFO2);
        printf("Fail to create FIFO3. Aborted.\n");
        return -1;
    }

    if((mknod(FIFO4, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        unlink(FIFO1);
        unlink(FIFO2);
        unlink(FIFO3);
        printf("Fail to create FIFO4. Aborted.\n");
        return -1;
    }
}
```



```
Activities Text Editor Thg 4 12 18:43
parent.c
~/Desktop/Lab5/S.2/Ex3

    unlink(FIFO2);
    printf("Fail to create FIFO3. Aborted.\n");
    return -1;
}

if((mknod(FIFO4, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
    unlink(FIFO1);
    unlink(FIFO2);
    unlink(FIFO3);
    printf("Fail to create FIFO4. Aborted.\n");
    return -1;
}

if((writefd=open(FIFO1, 1)) < 0)
    perror("Child cannot open writeFIFO.\n");

int c = atoi(argv[1]);
write(writefd, &c, sizeof(c));

if((writefd=open(FIFO2, 1)) < 0)
    perror("Child cannot open writeFIFO.\n");

int d = atoi(argv[2]);
write(writefd, &d, sizeof(d));

if((writefd=open(FIFO3, 1)) < 0)
    perror("Child cannot open writeFIFO.\n");

char tempString[1];
strcpy(tempString, argv[3]);
char tempOperator = tempString[0];
write(writefd, &tempOperator, sizeof(tempOperator));

if((readfd=open(FIFO4, 0)) < 0)
    perror("Child cannot open readFIFO.\n");

int temp;
read(readfd, &temp, sizeof(temp));

printf("%d %c %d = %d\n", c, tempOperator, d, temp);
close(readfd);
close(writefd);
return 1;
}
```

Bracket match found on line: 17

```
Activities Text Editor Thg 4 12 18:42
child.c
~/Desktop/Lab5/5.2/Ex3 Save

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <fcntl.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define FIFO3 "/tmp/ff.3"
#define FIFO4 "/tmp/ff.4"
#define PM 0666
extern int errno;
#define PIPE_BUF 4096

int calculator(int a, int b, char operator) {
    switch(operator) {
        case '+':
            return a + b;
        case '-':
            return a - b;
        case '*':
            return a * b;
        case '/':
            return a / b;
    }
}

int main(int argc, char* argv[]) {
    int a, b;
    char operator;
    int readfd, writefd;

    if((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        printf("Fail to create FIFO1. Aborted.\n");
        return -1;
    }

    if((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        unlink(FIFO1);
        printf("Fail to create FIFO2. Aborted.\n");
        return -1;
    }
}
```

```
Activities Text Editor Thg 4 12 18:42
child.c
~/Desktop/Lab5/5.2/Ex3 Save

int main(int argc, char* argv[]) {
    int a, b;
    char operator;
    int readfd, writefd;

    if((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        printf("Fail to create FIFO1. Aborted.\n");
        return -1;
    }

    if((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        unlink(FIFO1);
        printf("Fail to create FIFO2. Aborted.\n");
        return -1;
    }

    if((mknod(FIFO3, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        unlink(FIFO1);
        unlink(FIFO2);
        printf("Fail to create FIFO3. Aborted.\n");
        return -1;
    }

    if((mknod(FIFO4, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
        unlink(FIFO1);
        unlink(FIFO2);
        unlink(FIFO3);
        printf("Fail to create FIFO4. Aborted.\n");
        return -1;
    }

    if((readfd=open(FIFO1, 0)) < 0)
        perror("Child cannot open readFIFO.\n");

    read(readfd, &a, sizeof(a));

    if((readfd=open(FIFO2, 0)) < 0)
        perror("Child cannot open readFIFO.\n");

    read(readfd, &b, sizeof(b));

    if((readfd=open(FIFO3, 0)) < 0)
        perror("Child cannot open readFIFO.\n");
}
```

```
Activities Text Editor Thg 4 12 18:43
~/Desktop/Lab5/5.2/Ex3 child.c
Save

}
printf("Fail to create FIFO2. Aborted.\n");
return -1;

}

if((mkfifo(FIFO3, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
    unlink(FIFO1);
    unlink(FIFO2);
    printf("Fail to create FIFO3. Aborted.\n");
    return -1;
}

if((mkfifo(FIFO4, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)) {
    unlink(FIFO1);
    unlink(FIFO2);
    unlink(FIFO3);
    printf("Fail to create FIFO4. Aborted.\n");
    return -1;
}

if((readfd=open(FIFO1, 0)) < 0)
    perror("Chld cannot open readFIFO.\n");

read(readfd, &a, sizeof(a));

if((readfd=open(FIFO2, 0)) < 0)
    perror("Chld cannot open readFIFO.\n");

read(readfd, &b, sizeof(b));

if((readfd=open(FIFO3, 0)) < 0)
    perror("Chld cannot open readFIFO.\n");

read(readfd, &operator, sizeof(operator));

if((writefd=open(FIFO4, 1)) < 0)
    perror("Chld cannot open writeFIFO.\n");

int res = calculator(a, b, operator);
write(writefd, &res, sizeof(res));

close(readfd);
close(writefd);
return 1;
}

Bracket match found on line: 30 C Tab Width: 8 Ln 86, Col 2 INS
```

```
Activities Terminal Thg 4 12 18:46
asus@asus-virtual-machine: ~/Desktop/Lab5/5.2/Ex3
asus@asus-virtual-machine:~$ cd Desktop/Lab5/5.2/Ex3
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ gcc -c parent.c
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ gcc -o parent.out parent.o
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ ./parent.out 6 4 +

asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$
asus@asus-virtual-machine:~$ cd Desktop/Lab5/5.2/Ex3
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ gcc -c chld.c
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ gcc -o chld.out chld.o
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$
```

```
Activities Terminal Thg 4 12 18:46
asus@asus-virtual-machine: ~/Desktop/Lab5/5.2/Ex3
asus@asus-virtual-machine:~$ cd Desktop/Lab5/5.2/Ex3
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ gcc -c parent.c
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ gcc -o parent.out parent.o
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ ./parent.out 6 4 +
6 + 4 = 10
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$

asus@asus-virtual-machine:~$ cd Desktop/Lab5/5.2/Ex3
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ gcc -c chld.c
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ gcc -o chld.out chld.o
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ ./chld.out
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$
```

```
Activities Terminal Thg 4 12 18:48
asus@asus-virtual-machine: ~/Desktop/Lab5/5.2/Ex3
asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$

asus@asus-virtual-machine:~/Desktop/Lab5/5.2/Ex3$ ./chld.out
```

